Assignment-3

```
#include <iostream>
#include <GL/glut.h>
#include <math.h>
using namespace std;
int cx=300,cy=300,R=70;
bool flag=1;
struct color{
GLubyte r,g,b;
};
void init()
{
glClearColor(1,1,1,0);
glClear(GL_COLOR_BUFFER_BIT);
gluOrtho2D(0,600,0,600);
glColor3f(0,0,0);
}
void plotpixel(int x,int y)
{
glPointSize(1.5);
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
glFlush();
}
void octant(int xc,int yc,int x,int y)
{
plotpixel(xc+x,yc+y);
plotpixel(xc+y,yc+x);
plotpixel(xc+y,yc-x);
```

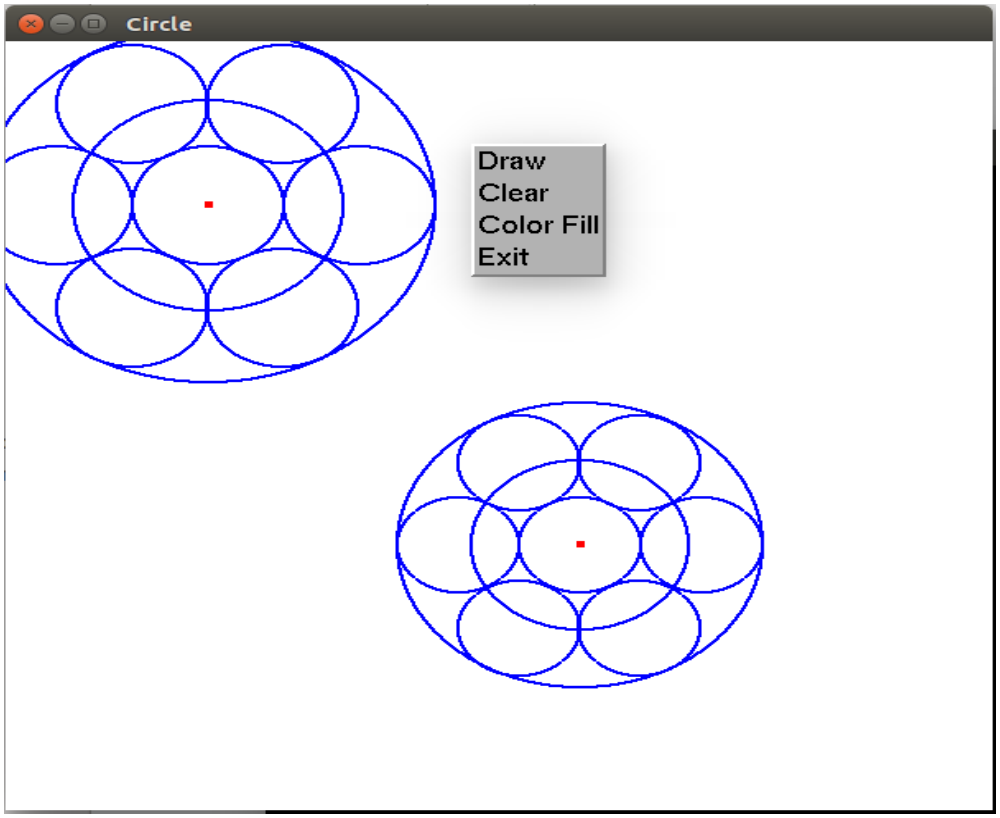
```
plotpixel(xc+x,yc-y);
plotpixel(xc-x,yc-y);
plotpixel(xc-y,yc-x);
plotpixel(xc-y,yc+x);
plotpixel(xc-x,yc+y);
}
void circleMP(int xc,int yc,int r)
{
int p=1-r,x=0,y=r;
while(x<y)
{
octant(xc,yc,x,y);
x++;
if(p>0)
y--,p+=2*(x-y)+1;
else
p+=2*x+1;
}
}
double ang(int q)
{
return (double)q*3.142/180;
}
void plottofill(int x,int y,color c)
{
glPointSize(1.0);
glColor3ub(c.r,c.g,c.b);
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
glFlush();
```

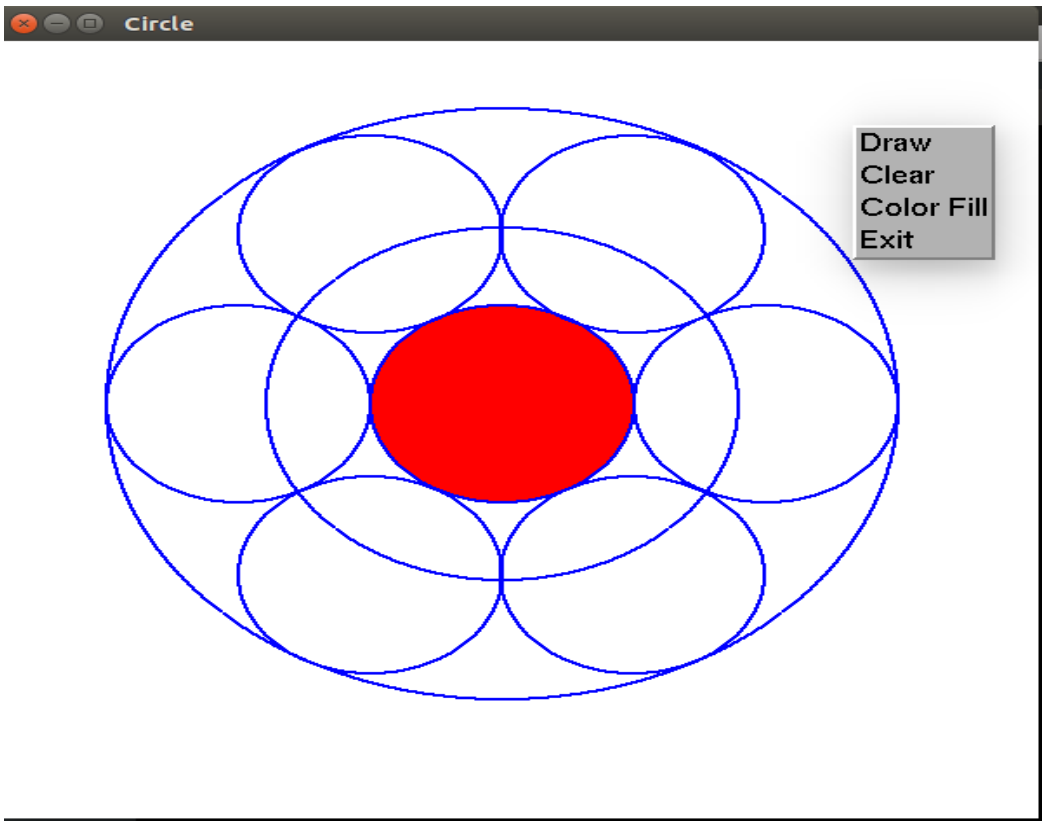
```
}
void seedfill(int x,int y,color oc,color nc)
{
color c;
glReadPixels(x,y,1,1,GL_RGB,GL_UNSIGNED_BYTE,&c);
if(c.r==oc.r&&c.b==oc.b&&c.g==oc.g)
{
plottofill(x,y,nc);
seedfill(x+1,y,oc,nc);
seedfill(x-1,y,oc,nc);
seedfill(x,y+1,oc,nc);
seedfill(x,y-1,oc,nc);
}
}
void drawcircles(int x,int y,int r)
{
circleMP(x,y,r);
circleMP(x+2*r,y,r);
circleMP(x-2*r,y,r);
circleMP(x+2*r*cos(ang(60)),y+2*r*sin(ang(60)),r);
circleMP(x-2*r*cos(ang(60)),y+2*r*sin(ang(60)),r);
circleMP(x-2*r*cos(ang(60)),y-2*r*sin(ang(60)),r);
circleMP(x+2*r*cos(ang(60)),y-2*r*sin(ang(60)),r);
circleMP(x,y,3*r);
circleMP(x,y,(float)2*r-r*(0.20));
}
void draw()
{

}
```

```
void clear_screen()
{
glClearColor(1,1,1,0);
glClear(GL_COLOR_BUFFER_BIT);
}
void mouseClick(int button,int state,int x,int y)
{
cout<<"Mouse Clicked"<<endl;
//First point to get the xc,yc
if(flag&&button==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)
{
cout<<"Center Found"<<endl;
cx=x,cy=600-y;
glPointSize(5.0);
glColor3f(1,0,0);
glBegin(GL_POINTS);
glVertex2i(x,600-y);
glEnd();
glFlush();
flag=0;
}
else if (!flag&&button==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)
{
cout<<"Ohhho !!, I got a radius"<<endl;
glColor3f(0,0,1);
glPointSize(1.0);
glBegin(GL_POINTS);
glVertex2i(x,600-y);
glEnd();
glFlush();
R=abs(x-cx);
```

```
flag=1;
}
}
void menu(int ch)
{
color oc={255,255,255};
color nc={255,0,0};
switch(ch)
{
case 1:
drawcircles(cx,cy,R);
break;
case 2:
clear_screen();
break;
case 3:
cout<<"Fill the Centered Circle"<<endl;
seedfill(cx+5,cy,oc,nc);
break;
case 4:
exit(0);
break;
}
}
int main(int agrc,char ** agrv)
{
glutInit(&agrc,agrv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowPosition(0,0);
glutInitWindowSize(600,600);
glutCreateWindow("Circle");
```

```
init();
glutDisplayFunc(draw);
glutCreateMenu(menu);
glutAddMenuEntry("Draw",1);
glutAddMenuEntry("Clear",2);
glutAddMenuEntry("Color Fill",3);
glutAddMenuEntry("Exit",4);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutMouseFunc(mouseClick);
glutMainLoop();
}
```


Circle
Draw
Clear
Color Fill
Exit


Circle
Draw
Clear
Color Fill
Exit

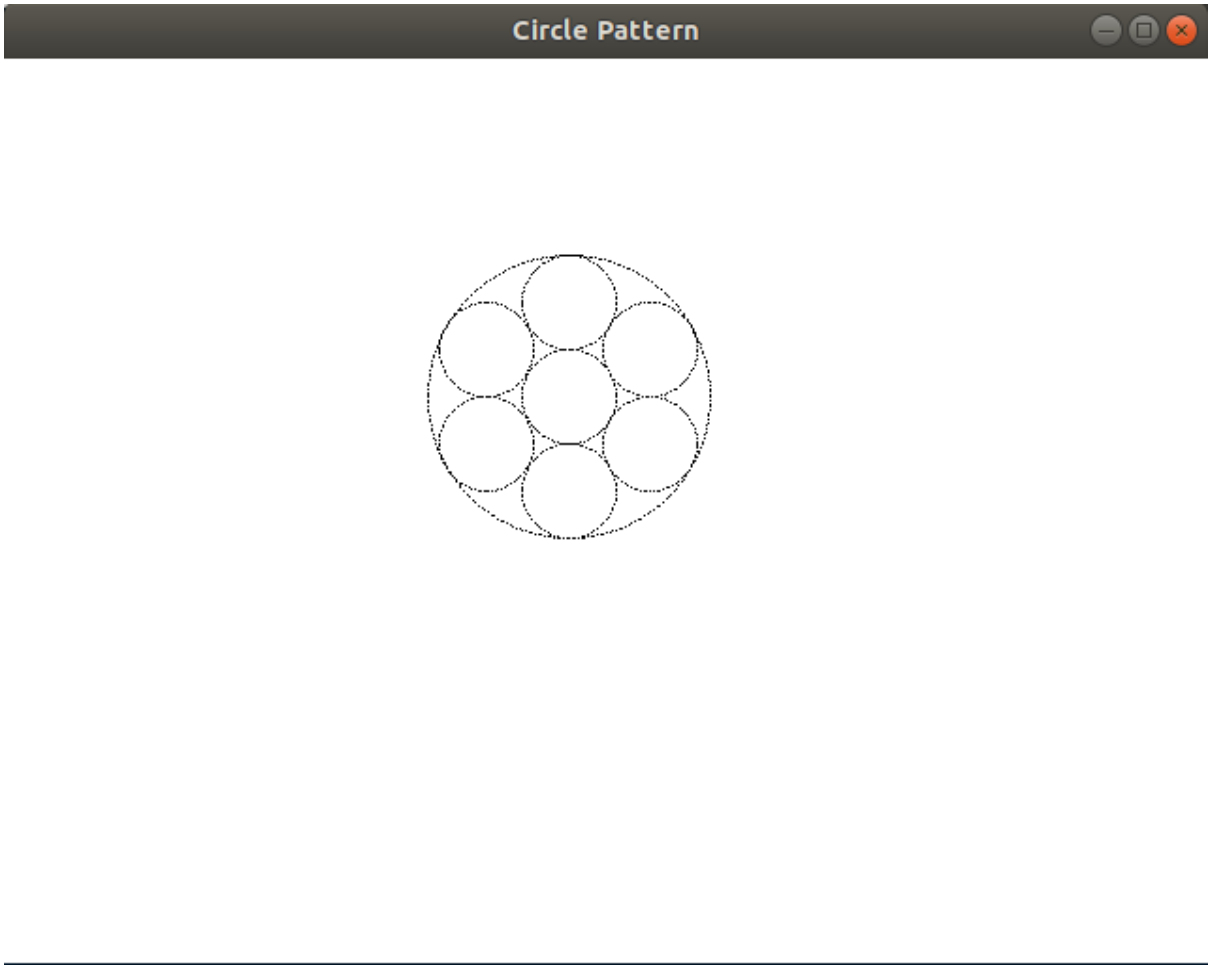
```
#include<stdio.h>                       //initial inclusions
#include<GL/gl.h>
#include<GL/glu.h>
#include<GL/glut.h>
#include<math.h>
int xc,yc,xo,yo,r;
static int p=0;
void setpoint(int x,int y)                      //setting point on the window
{
if(p>2)
{
if(p==4)
p=0;
else
p++;
}
else
{
glColor3f(0.0,0.0,0.0);
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
p++;
}
}
void plotpoint(int x,int y)  //plotting points in all octants
{
setpoint(xc+x,yc+y);
setpoint(xc-x,yc+y);
setpoint(xc+x,yc-y);
setpoint(xc-x,yc-y);
```

```
setpoint(xc+y,yc+x);
setpoint(xc-y,yc+x);
setpoint(xc+y,yc-x);
setpoint(xc-y,yc-x);
}
void midpoint(int rad)                    //midpoint circle drawing algorithm
{
int x=0,y=rad,p=1-rad;
plotpoint(x,y);                           //plotting initial point
while(y>x)                                //iterating till y>x
{
x=x+1;                                    //incrementing x
if(p>0)                                   //checking condition for p
{
y--;                                      //decrementing y
p+=2*(x-y)+1;                             //changing value of p
}
else
{
p+=2*x+1;                                 //changing value of p
}
plotpoint(x,y);
}

}
void mouse(int btn,int state,int x,int y)
{
static int q=1;
if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)//checking left click
{
switch(q)
```

```
{
 case 1:midpoint(r);
q++;
break;
case 2:yc=yc+2*r;
midpoint(r);
q++;
break;
case 3:yc=yo;
yc=yc-2*r;
midpoint(r);
q++;
break;
case 4:xc=xo;
yc=yo;
xc=xc+2*r*0.866;
yc=yc+r;
midpoint(r);
q++;
break;
case 5:xc=xo;
yc=yo;
xc=xc+2*r*0.866;
yc=yc-r;
midpoint(r);
q++;
break;
case 6:xc=xo;
yc=yo;
xc=xc-2*r*0.866;
yc=yc-r;
```

```
midpoint(r);
q++;
break;
case 7:xc=xo;
yc=yo;
xc=xc-2*r*0.866;
yc=yc+r;
midpoint(r);
q++;
break;
case 8:xc=xo;
yc=yo;
midpoint(3*r);
q++;
break;
}
glFlush();
}
}
void init()
{
glClearColor(1.0,1.0,1.0,0);                    //clearing background color to new color
glClear(GL_COLOR_BUFFER_BIT);                   //clearing buffer
gluOrtho2D(0,640,0,480);                        //decalring ortho 2d coordinates
glPointSize(1);
glFlush();
}
int main(int argc,char **argv)
{
printf("Enter coordinates of centre of circle\n");
printf("\nX: ");
```

```
scanf("%d",&xc);
printf("\nY: ");
scanf("%d",&yc);
xo=xc;
yo=yc;
printf("\nEnter radius of circle:");
scanf("%d",&r);
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowPosition(100,100);
glutInitWindowSize(640,480);
glutCreateWindow("Circle Pattern");
init();
//glutDisplayFunc(disp);
glutMouseFunc(mouse);
glutMainLoop();
return 0;
}
```


Circle Pattern